

# Rweibo使用说明

Jian Li\*

## 1 注册应用

### 1.1 简介

Rweibo 是一个 R 语言下操作新浪微博的包，基于新浪提供的API进行开发，实现了大部分的接口函数。调用新浪API的时候，需要通过OAuth2.0的方式进行授权，因此每位用户登录自己的微博帐号之后，还需要申请一个自己的应用，得到该应用的App Key和App Secret，然后利用Rweibo包实现OAuth2.0的授权，从而通过 R 中的函数调用新浪提供的接口，在自己的微博帐号下进行各种操作。

### 1.2 在新浪开放平台中创建应用

使用某个新浪微博的帐号登录，进入“新浪微博开放平台”(<http://open.weibo.com/apps>)，可以创建应用。在本版的API中，普通用户只能创建网页应用，所以当前版本的 Rweibo 包是基于网页API开发的。

选择网页应用后，在创建新应用的界面输入自己应用的详细信息，然后提交申请，如果信息填写完备，则会申请成功，会得到该应用的App Key和App Secret。

详细的创建应用、申请权限的过程请参考<http://jliblog.com/app/rweibo> 上的文档《新浪微博开放平台权限申请指南》。

### 1.3 OAuth2.0 授权设置

在最新的新浪API下，使用了OAuth 2.0，普通权限只能申请网页应用，因此需要对“授权回调页”和“取消授权回调页”进行设置，在“授权回调页”处输入Rweibo包中默认的<http://127.0.0.1/library/Rweibo/doc/callback.html>，注意，此处的端口号（默认为80）可以进行更改，其他的部分最好不要修改，除非真的明白其中的含义并且已准备好自己的回调页（可以放在自己的网站上）。

---

\*新浪微博: lijian001

如果需要修改默认的端口号（比如端口80和已存在的某个服务冲突），将以上回调页的URL中的端口号换成自己自定的数字，同时切记要在Rweibo包的安装路径下将端口号也改成这个数字：打开 %R\_HOME%\library\Rweibo\config\Rweibo.txt，将port之后的数字改成新的端口号。

注意，在当前版本的微博API中，回调页的URL中如果带有端口号，则会提示格式错误。所以最好不要修改端口号，除非API恢复了之前支持端口号的功能。

另外“取消授权回调页”可以不用填写。

## 2 在R中管理应用

### 2.1 打开R环境

启动R环境，可以直接打开RGui，也可以在R的IDE（比如Eclipse、RStudio）中启动R，然后加载Rweibo包。如果之前没有安装过Rweibo包，需要安装后才能使用。当前版本在Rforge（[https://r-forge.r-project.org/R/?group\\_id=1054](https://r-forge.r-project.org/R/?group_id=1054)）进行维护，可以直接安装最新版。如果是老版的R或者提示“package ‘Rweibo’ is not available”，则应使用“source”的方式进行安装。

```
1 > install.packages("Rweibo", repos="http://R-Forge.R-project.org")
2 > #install.packages("Rweibo", repos="http://R-Forge.R-project.org", type="source")
```

如果Rweibo已经安装成功，加载该包：

```
1 > require(Rweibo)

1 Loading required package: Rweibo
2 Loading required package: tools
3 Loading required package: RCurl
4 Loading required package: bitops
5 Loading required package: rjson
6 Loading required package: XML
7
8 Attaching package: ‘XML’
9
10 The following object(s) are masked from ‘package:tools’ :
11
12     toHTML
13
14 Loading required package: digest
15 # Rweibo Version: 0.2-7
```

可以看到当前的Rweibo包的版本。

## 2.2 管理应用

对于在新浪开发平台中申请的应用，可以利用Rweibo将其存成R对象，在R中进行管理。

比如该应用的App Key为“GDdmIQH6jh”，App Secret为“MCD8BKwGdgPHv”，我们在R中给其起名为“mytest”，首先在R中注册该应用：

```
1 > registerApp(app_name = "mytest", "GDdmIQH6jh", "MCD8BKwGdgPHv")
```

查看该应用：

```
1 > listApp("mytest")
```

```
1 $app_key
2 [1] "GDdmIQH6jh"
3
4 $app_secret
5 [1] "MCD8BKwGdgPHv"
6
7 $app_token
8 list()
```

修改该应用，比如App Secret被重置成了“KSDk0wwH7tRep”：

```
1 > modifyApp(app_name = "mytest", "GDdmIQH6jh", "KSDk0wwH7tRep")
2 > listApp("mytest")
```

```
1 $app_key
2 [1] "GDdmIQH6jh"
3
4 $app_secret
5 [1] "KSDk0wwH7tRep"
6
7 $app_token
8 list()
```

该应用可以被删除：

```
1 > deleteApp("mytest")
```

```
2 > listApp("mytest")
```

```
1 Error in listApp("mytest") :
```

```
2 mytest doesn't exist, please use 'registerApp' to create
```

### 3 对应用进行授权

#### 3.1 OAuth授权机制简介

旧版的新浪微博API以及Rweibo包（ $\leq 0.1$ ）使用的是OAuth 1.0。对于OAuth 1.0的授权过程，简单地来说，开发者希望能开发一个应用，让不同的微博用户使用，每个用户都有自己的微博帐号信息，微博用户不希望在使用的过程中被开发者知道，同时开发者的App Key和App Secret也不希望被别的用户知道。这样一来，就需要一种授权机制，使得开发者和微博用户都信得过，这就需要基于一个可信的第三方平台。对于新浪微博来说，双方都信任新浪，只要能确认所处的网站是真的新浪微博，就不会有问题。先是开发者将自己的App Key和App Secret发送到一个请求地址，新浪微博验证无误后，返回请求的Token，其实相当于是加密过的开发者信息，将这个信息发送到授权地址，可以完成用户的授权。用户在授权的页面可以输入自己的微博帐号，这些信息只有新浪知道，然后会返回一个验证码，用户输入这个验证码之后进行再次确认。开发者将之前自己的Token信息以及用户的这个验证码一起发送到一个访问地址，经过确认后，新浪微博觉得双方都是可信的，而且彼此也都不知道对方的信息，就会返回一个访问的Token以及密码，这就好比是一把钥匙，开发者只要存着这把钥匙，下次如果再对该用户做某些操作时就不需要再征求他的同意了，除非用户主动废除这把钥匙。以上的过程就是OAuth授权的过程。

在目前的OAuth 2.0中，原理没有变化，但是授权的过程变得简单了很多。开发者将自己的App Key以及授权回调页（通常是放在自己站点上的一个动态页，在Rweibo中采用内置的本机html文件）信息做成一个基于新浪的链接发给微博用户，用户如果能确定该链接来自新浪，就可以用自己的帐号和密码进行授权，新浪会返回一个处理后的授权Token给开发者，开发者利用这个授权Token再加上自己的App Secret信息向新浪获取一个访问Token，以后就可以利用这个Token对其绑定的应用及微博用户进行各项操作。

当前版本的新浪API的访问Token的有效期只有一天，也就是说24小时之后，需要重新进行一次授权过程（文档注明的有效期是24小时，但是实际使用中可能会超过24小时），比起上一版的API，变麻烦了一些，不过对于R用户来说，主要目的是分析，影响不是很大。

#### 3.2 OAuth授权过程

在R中，最后得到的访问Token存于一个R对象中，比如：

```
1 > roauth <- createOAuth(app_name = "mytest", access_name = "rweibo")
```

roauth 将是一个储存了所有授权信息的R对象，作为后续所有函数的一个输入参数。在该函数中，app\_name 表示一个已存在的应用的名称，access\_name 为针对某个帐号授权的名称，一般使用该帐号的昵称（可以任意起名，但不支持中文名）。

默认在新建授权对象时对当前登录的用户进行OAuth授权，表示可以使用API的方式对新浪微博进行访问。从Rweibo 0.2-1开始，授权对象还能支持保存Cookie的授权方式，通过模拟登录来对新浪微博进行访问，在这种情况下可以无需注册API和建立OAuth授权，但是只能使用“web.”开头的接口。详细信息参见3.3。

OAuth授权的过程中，执行完 createOAuth信息后，将会自动弹出一个授权页，如果需要重新登录，第一个页面是应用授权的登录页，见图1。



图 1: 应用授权页

如果帐号信息无误，则会得到一个返回的包含CODE的授权回调页，“CODE:”后面的那一串字符就是系统返回的授权信息，如图2所示。

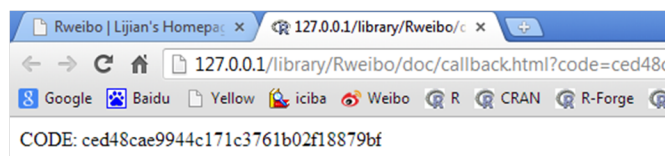


图 2: 授权回调页

```
> roauth <- createOAuth("mytest", "rweibo", forclogin = TRUE)
Please input the codes here
CODE: db433ac7b69e14752fe1b6ec10c03d64|
```

图 3: R语言授权

将这串字符串复制到剪贴板，回到R的界面，可以看到R中有个等待输入的区域（图3），将刚才复制到的CODE字符串粘贴在此处然后回车，将会完成一个授权。

在该授权的有效期内（24小时），即使重启过R，只需再次运行：

```
1 > roauth <- createOAuth(app_name = "mytest", access_name = "rweibo")
```

就可以自动读取保存后的授权信息，无需再次授权即可得到一个R中的授权对象roauth。

### 3.3 模拟登录的授权

新浪提供的官方API（OAuth的方式授权）可以对微博进行所有的操作。但是受到普通用户的权限限制，抓取信息的频率被严格控制，此外有些高级权限的API（比如内容搜索）也无法使用。因此，从Rweibo 0.2-1开始，Rweibo包将会陆续加入通过解析网页的方式获取信息的接口。也提供了模拟登录并保存Cookies的方式作为补充。

从2013年7月2日开始，新浪对接口进行了调整，获取他人微博、粉丝列表、关注列表的功能对于普通权限接口已经失效，其他接口不受影响（抓评论转发等），高级的权限下接口也不受影响。针对新浪的这次调整，本包从Rweibo 0.2-7版开始，正式引入了模拟登录的方式。通过模拟网页登陆，输入用户的帐号和密码信息，然后保存Cookies，在登录的状态下对网页进行抓取和解析，从而获取到相应的信息。

模拟登录的方式参考了波波头一头的weiBor (<https://github.com/yibochen/weiBor>) 项目<sup>1</sup>。并针对新浪新的rsa加密方式进行了改写。当前的登录脚本是v1.4.5，由于新浪可能随时更新登录脚本，旧的版本就会失效，请随时联系作者<sup>2</sup>进行修改。

在创建授权对象的时候，可以选择“模拟登录”（login = TRUE），需要输入某个微博账户的用户名和密码，例如：

```
1 > roauth <- createOAuth(app_name = "mytest", access_name = "rweibo",  
2   login = TRUE, username = "myaccount", password = "mypwd")
```

也可以在创建成功的对象中运行login方法进行登录：

```
1 > roauth$login(username = "myaccount", password = "mypwd")
```

需要注意的是，这种方式会在本机保存Cookies，在使用的时候请确认不存在安全隐患，否则不推荐使用这种授权方式。另外，由于用户名和密码都是使用明文的方式传入R，如果不小心的话，容易被他人获取自己的密码。所以，对于这种模拟登录的方式，最好不要使用自己的主帐号，用马甲即可（模拟登录的帐号与OAuth授权的帐号可以不一致）。

对于初学者，如果仅仅是进行尝试，官方API的功能基本足够（目前只是无法抓取他人的微博内容、粉丝列表和关注列表，根据微博ID抓取评论和转发的功能仍然不受影响）。缺少的内容搜索，也可以使用“web.search.content”函数来实现，注意，这个内容

---

<sup>1</sup>该项目下还包含一些微博分析的函数。

<sup>2</sup>通过新浪微博联系响应最快，帐号:lijian001

搜索的函数**不要求任何授权!** 即使不建立授权对象, 也可以使用, 详细的说明请查看该函数的文档。

目前通过模拟登录保存的Cookies来进行网页信息抓取的函数包括web.user\_timeline, 详情请参考5.4的例子。

## 4 授权后的weibo2.0对象

授权后的对象为一个类为weibo2.0的对象:

```
1 > class(roauth)

1 [1] "weibo2.0"
2 attr(,"package")
3 [1] "Rweibo"
```

### 4.1 信息查看

直接在控制台输入授权对象名然后回车可以查看基本信息, 如果未进行模拟登录或者保存Cookies, 则只显示oauth的授权信息, 例如:

```
1 > roauth

1 [1] "Application: rweibodemo (328007540)"
2 [1] "Access: rdemo (1318558807)"
3 [1] "oauth was authorized! (expires in 30.73 hours)"
```

如果已经进行过OAuth授权, 则会显示应用名(以及应用的Key)和授权帐号名称(以及帐号的UID), 并注明“oauth was authorized!”, 还会显示过期的时间(注意每次查看时过期时间会不同)。

如果已经成功进行过模拟登录, 则会显示登录的信息, 例如:

```
1 [1] "Application: rweibodemo (328007540)"
2 [1] "Access: rdemo (1318558807)"
3 [1] "oauth was authorized! (expires in 30.73 hours)"
4 [1] "cookies were saved! (rdemo.cookies)"
```

## 4.2 查看API权限的限制

官方API针对权限的不同，每小时能操作的频率也有不同。为了能实时地了解当前小时（或天）内的剩余权限，可以使用getLimits函数。

```
1 > roauth$getLimits(TRUE)

1 Reset time: 2013-05-26 21:00:00
2
3      API LimitUnit Limit RemainingHits
4 1    /statuses/update    HOURS    30          30
5 2    /comments/create    HOURS    60          60
6 3 /direct_messages/new    HOURS    60          60
7 4  /friendships/create    HOURS    60          60
8 5  /friendships/create    DAYS    100         100
9 6            ip_limit    HOURS  1000         1000
7 7            user_limit    HOURS   150          150
```

该函数只有一个参数，默认为FALSE，表示通过Rweibo提供的记录器。TRUE表示使用新浪提供的权限查看API。无论是Rweibo的记录器还是新浪的API，使用getLimits函数时都不会影响剩余的请求数（也就是说，查看权限的操作并不会使得剩余总操作数减1）。

## 5 函数调用

### 5.1 查看微博

查看公共微博：

```
1 > res1 <- statuses.public_timeline(roauth, count = 5)
2 > res1[[1]]
```

查看自己以及关注对象的微博：

```
1 > res2 <- statuses.friends_timeline(roauth, count = 5)
2 > res2[[1]]
```

查看双向关注的好友的最新微博：

```
1 > res3 <- statuses.bilateral_timeline(roauth, count = 5)
2 > res3[[1]]
```

查看某位微博用户的最新微博：



```
1 > res4 <- statuses.user_timeline(roauth, screen_name = "Rweibo", count = 5)
2 > res4[[1]]
```

查看当前用户收到的评论:

```
1 > res5 <- comments.to_me(roauth, count = 5)
2 > res5[[1]]
```

查看@了当前用户的最新微博:

```
1 > res6 <- statuses.mentions(roauth, count = 5)
2 > res6[[1]]
```

查看@了当前用户的评论

```
1 > res7 <- comments.mentions(roauth, count = 5)
2 > res7[[1]]
```

## 5.2 发表微博

发表一条微博:

```
1 > res8 <- statuses.update(roauth, status = "你好啊*!@#$$&=+")
```

转发某条微博:

```
1 > res9 <- statuses.repost(roauth, id = res8$idstr, status = "转一个啊")
```

评论某条微博:

```
1 > res10 <- comments.create(roauth, id = res9$idstr, comment = "评论一下啊")
```

## 5.3 搜索微博

当前的新浪API取消了普通用户的搜索接口, 所以无法通过OAuth的方式实现对微博内容的搜索。因此此包的作者自行开发了通过web方式(以 web. 开头的接口)的搜索接口。这些接口通过XML方式解析Web内容来实现, 无需建立 OAuth2 对象。优点是可以不用占用API的访问权限。缺点是Web页的格式有可能会发生改变, 如果用户在使用过程中发现解析出来的内容出现问题, 请联系作者(新浪微博:lijian001)进行修改。

```
1 > res11 <- web.search.content("R语言", page = 2, combinewith = NULL)
```

第一个参数 `sword` 代表搜索的关键词，`page` 参数表示需要获取的页面数，默认为1，最大值为50（由于新浪微博对过于频繁的抓取会采取暂时封禁IP的处理，经过作者的试验，每次抓取25页以内是比较安全的方式，从0.2-1版开始，任何超过25页的抓取都会自动减到25页）。每一页可以获取20条微博。如果搜索过于频繁，有可能会被封，因此可以设置 `sleepmean` 和 `sleepsd` 参数为正值（默认都为0，表示产生正态随机数的均值和标准差）。

`combinewith` 参数可以实现增量搜索，默认为 `NULL`，表示重新搜索。如果将其设为某个之前的结果集，则可返回增量搜索的结果：

```
1 > res12 <- web.search.content("R语言", page = 2, combinewith = res11)
```

参数 `sinceID` 表示搜索某个微博ID之后的微博，默认为 `NULL`。

```
1 > res13 <- web.search.content("Rweibo", sinceID = "3508307023192146")
```

参数 `since` 表示搜索某个时间之后的微博，需要为 `POSIXlt` 对象。如果输入字符串的话，必须是“YYYY-MM-DD”格式的时间。

```
1 > res14 <- web.search.content("Rweibo", since = "2012-10-01")
```

最后返回的结果包含八列：

```
1 > names(res11)

1 [1] "MID"           "Author"        "Weibo"         "Forward"
2 [5] "Time_Weibo"    "Time_Search"   "Count_Forward" "Count_Reply"
```

分别表示微博ID、微博作者、微博内容、转发的原帖的内容、微博发布时间、该次搜索的时间、转发数、评论数。

该接口也支持多个关键词的查询。空格“ ”表示“和”，“~”符号表示“或”，例如：

```
1 > res15 <- web.search.content("Rweibo lijian001", page = 1, combinewith = NULL)
2 > res16 <- web.search.content("Rweibo~lijian001", page = 1, combinewith = NULL)
```

## 5.4 模拟登录授权

2013年7月2日开始，新浪取消了部分API的功能，从Rweibo 0.2-7开始，加入了通过模拟登录保存Cookies的方式进行授权的接口，这些接口都以 `web.` 开头，需要传入经过了模拟登录授权后的 `weibo2.0` 对象。

`web.user_timeline` 可以用来获取除自己外的其他用户的微博：

```

1 u17 <- web.search.user("lijian001")
2 res17 <- web.user_timeline(roauth, uid = u1$uid, pages = 1:2)

```

最后返回的结果包含十列：

```

1 > names(res17)

1 [1] "MID"          "Author"       "UID"          "Weibo"        "Forward"
2 [6] "Time_Weibo"   "Time_Search"  "Count_Forward" "Count_Reply"  "Client"

```

## 5.5 通用api

从0.2-5版开始，新加入了weibo.api 这一通用API。对于暂未引入本包或者新浪接口有变的接口，可以使用这个通用的函数自行定义接口。

我们以public\_timeline 为例，假设我们并没有statuses.public\_timeline 这个函数。首先进入该接口的API文档页面：[http://open.weibo.com/wiki/2/statuses/public\\_timeline](http://open.weibo.com/wiki/2/statuses/public_timeline)。

weibo.api的第一个参数是之前创建的roauth对象，第二个参数URL就是这个API的URL，可以在该文档页面查到json接口的URL：[https://api.weibo.com/2/statuses/public\\_timeline.json](https://api.weibo.com/2/statuses/public_timeline.json)。

第三个参数paramlist 是一个列表，在文档页面可以看到“请求参数”这个表格，除去source和access\_token 之外的其他参数都可以输入到这里，以表格中出现的参数名为列表的成分名。比如list(count = 5)。

第四个参数httpmethod 是一个字符，可以选择“GET”或者“POST”，对应文档页面中“HTTP请求方式”的值。

在这个例子下，我们可以直接通过weibo.api 的方式来调用这个接口：

```

1 > res18 <- weibo.api(roauth,
2 + URL = "https://api.weibo.com/2/statuses/public_timeline.json",
3 + paramlist = list(count = 5, page = 1),
4 + httpmethod = "GET")
5 >

```

## 6 分析接口

在新浪提供的API之上，本包自0.2-3版开始提供以analysis开头的分析接口。将一些常用的数据获取和分析的功能封装成函数。

## 6.1 获取转发信息

analysis.getReposts 接口可以获取某条微博的转发信息。例如:

```
1 ana1 <- analysis.getReposts(roauth, mid = "3575234466298494")
2 names(ana1)

1 [1] "created_at"          "mid"
2 [3] "text"                "reposts_count"
3 [5] "comments_count"      "attitudes_count"
4 [7] "in_reply_to_status_id" "in_reply_to_user_id"
5 [9] "in_reply_to_screen_name" "User_idstr"
6 [11] "User_screen_name"     "User_province"
7 [13] "User_city"            "User_location"
8 [15] "User_description"     "User_gender"
9 [17] "User_followers_count" "User_friends_count"
10 [19] "User_statuses_count" "User_favourites_count"
11 [21] "User_geo_enabled"     "User_created_at"
12 [23] "User_following"       "User_follow_me"
13 [25] "User_bi_followers_count" "User_verified"
14 [27] "User_verified_type"   "User_verified_reason"
```

该函数需要通过微博的mid号抓取，可以在网页上选择“查看网页源代码”，然后查找该微博中的某段文本，定位到该条微博，文本前方最近的“mid=”开头的一串16位的数字即为mid（形为“3575234466298494”）。有些浏览器查看源代码之后可能会显示UTF-8字符，无法通过复制文本来查找，可以尝试点击作者的昵称进入到作者页面，找到该微博的那一页，然后选择“查看网页源代码”，可能可以正常显示中文。

本函数默认查找所有的转发内容。但是该接口每查询一页将会用掉一次访问权限，而每一页只有200条微博的信息。对于普通权限的接口，一小时内最多只能查150次，也就是说，一次操作中最多只能获取30000条（由于其他的操作也可能会用掉权限，所以实际能抓取的数目通常会小于30000）转发信息。如果某条微博的转发量大于30000条，需要在一小时之内先抓取一部分（比如28000条），然后到了下一个小时之后再进行操作。比如说第一次想要查找前28000条，可以输入参数“count = 28000, startcount = 1”，第二次想查找第25000条到第30000条，可以输入参数“count = 5000, startcount = 25001”。

需要注意的是，某一页获取的微博总数可能和实际的不同，比如该页设置显示200条，实际抓取的数据中只有199条。这种情况的主要原因是某些微博设置了分组可见，如果不具备权限的话将无法获取该条微博。比如设置每页显示20条，发现第5页只获取了19条，我们从网页中进入转发信息的第5页，会发现该页只显示了19条，下方显示了“由于权限

设置，你无法查看所有转发内容。”的提示信息，而正常显示的第6页和第4页没有这个提示。

## 6.2 获取评论信息

`analysis.getComments` 接口可以获取某条微博的转发信息。例如：

```
1 ana2 <- analysis.getComments(roauth, mid = "3575234466298494")
2 names(ana2)

1 [1] "created_at"          "mid"
2 [3] "text"                "source"
3 [5] "User_idstr"          "User_screen_name"
4 [7] "User_province"       "User_city"
5 [9] "User_location"       "User_description"
6 [11] "User_gender"          "User_followers_count"
7 [13] "User_friends_count"   "User_statuses_count"
8 [15] "User_favourites_count" "User_geo_enabled"
9 [17] "User_created_at"      "User_following"
10 [19] "User_follow_me"       "User_bi_followers_count"
11 [21] "User_verified"        "User_verified_type"
12 [23] "User_verified_reason"
```

该函数的用法和`analysis.getReposts`完全相同。

## 6.3 获取某个用户的所有微博

`analysis.getUserTimeline` 接口<sup>3</sup>可以获取某条微博的转发信息。例如：

```
1 ana3 <- analysis.getUserTimeline(roauth, screen_name = "rweibo")
2 names(ana3)

1 [1] "created_at"          "mid"
2 [3] "text"                "reposts_count"
3 [5] "comments_count"      "attitudes_count"
4 [7] "in_reply_to_status_id" "in_reply_to_user_id"
5 [9] "in_reply_to_screen_name" "retweeted_created_at"
6 [11] "retweeted_mid"        "retweeted_text"
7 [13] "retweeted_reposts_count" "retweeted_comments_count"
```

---

<sup>3</sup>该接口目前只对高级权限有效。

```

8 [15] "retweeted_attitudes_count"      "retweeted_in_reply_to_status_id"
9 [17] "retweeted_in_reply_to_user_id"    "retweeted_in_reply_to_screen_name"
10 [19] "retweeted_user_idstr"             "retweeted_user_screen_name"
11 [21] "retweeted_user_province"          "retweeted_user_city"
12 [23] "retweeted_user_location"          "retweeted_user_description"
13 [25] "retweeted_user_gender"            "retweeted_user_followers_count"
14 [27] "retweeted_user_friends_count"     "retweeted_user_statuses_count"
15 [29] "retweeted_user_favourites_count"  "retweeted_user_geo_enabled"
16 [31] "retweeted_user_created_at"        "retweeted_user_following"
17 [33] "retweeted_user_follow_me"         "retweeted_user_bi_followers_count"
18 [35] "retweeted_user_verified"          "retweeted_user_verified_type"
19 [37] "retweeted_user_verified_reason"

```

参数uid和screen\_name必须输入一个，uid指的是用户ID，screen\_name指的是用户昵称。如果昵称是中文，需要将其转化成UTF-8编码。如果同时输入了uid和screen\_name，screen\_name将会被自动忽略。

count和startcount参数的用法与analysis.getReposts中相同，默认都是抓取该用户的所有微博。需要注意的是，用户微博接口每页最多只能抓取100条，也就是说，如果是普通权限，一小时之内能抓150次，最多只能获取该用户的15000条微博。

## 7 如何引用

BibTeX引用本文档：

```

@Manual{,
  author = {Jian Li},
  title = {Rweibo: An interface to the Weibo open platform},
  year = {2013},
  url = {http://jliblog.com/app/rweibo},
}

```